

École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
Fax +33 (0)2 47 36 14 22
www.polytech.univ-tours.fr

Spécialité Informatique Industrielle
4^{ème} année
2014-2015

Reconnaissance de grille

Projet de développement embarqué

Apprenti :
Alexandre BILLAY, Thibault ARTUS
alexandre.billay@etu.univ-tours.fr, theskull-machine@gmail.com

Tuteur :
Yannick KERGOSIEN
yannick.kergosien@univ-tours.fr
Polytech'Tours

Table des matières

Introduction	5
1 Cahier des charges	6
2 Conduite du projet	7
2.1 Chronogramme réel du projet	7
3 Recherches	8
3.1 Détection de formes	8
3.1.1 Transformée de Hough	8
3.1.2 Codage de Freeman absolu	9
3.1.3 Codage de Freeman relatif	9
3.1.4 Régression linéaire	10
3.2 Détection de couleur	10
4 Analyse des classes	11
4.1 Hough.java	11
4.1.1 Constructeur	11
4.1.2 Méthodes (Transformée de Hough)	11
4.1.3 Méthodes (Conversions)	12
4.1.4 Accesseurs	13
4.2 PictureHandler.java	13
4.2.1 Constructeur	13
4.2.2 Méthodes	13
4.3 HoughView.java	14
5 Tests effectués	15
6 Démonstration	16
Conclusion	17

Table des figures

2.1	Diagramme de Gantt	7
3.1	Transformée de Hough	8
3.2	Constitution d'une chaîne de codes	9
3.3	Codage du changement de direction	9
3.4	Nombre de directions	9

Introduction

Dans le cadre de notre quatrième année au sein de Polytech'Tours, nous avons dû réaliser un projet de développement embarqué sur une durée égale à 4 mois. Après 2 heures de présentation des sujets, nous avons choisi le développement et intégration d'un système de reconnaissance de grille sur tablette Android. Ce projet découle d'un PFE (Projet de Fin d'Étude), le Stack Crane Problem interprété par un pont roulant, réalisé lors l'année précédente par Thibault Morelle. Quant à notre projet, il fut décidé de reprendre la partie du PFE sur la reconnaissance automatique des objets à déplacer par simple prise de photo. Cette partie ne fonctionnait pas. Notre projet a dû être fait en collaboration avec Clément Laloubeyre, un élève en cinquième année qui a pris la suite du PFE de Thibault Morelle.

CHAPITRE 1

Cahier des charges

L'objectif du projet est de concevoir un module de détection d'une grille et de localisation de deux types de pièces de couleurs dans cette grille à l'aide de la caméra d'une tablette Android. Ce module sera à intégrer dans une application mobile permettant de contrôler un pont roulant ayant pour but de déplacer les objets détectés dans cette grille.

CHAPITRE 2

Conduite du projet

2.1 Chronogramme réel du projet

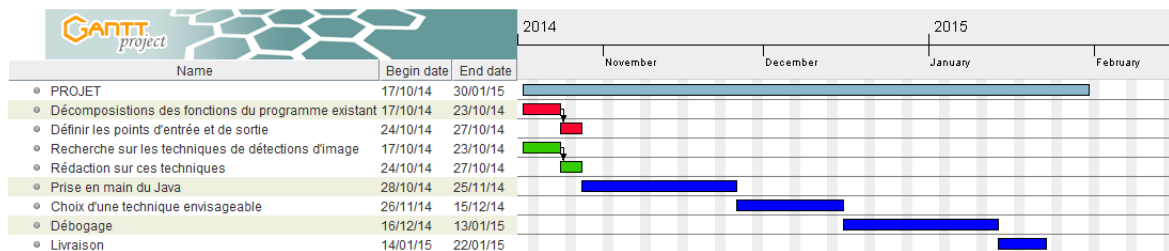


FIGURE 2.1 – Diagramme de Gantt

CHAPITRE 3

Recherches

3.1 Détection de formes

3.1.1 Transformée de Hough

Le principe qui sous-tend la transformée de Hough est qu'il existe un nombre infini de lignes qui passent par un point, dont la seule différence est l'orientation (l'angle). Le but de la transformée est de déterminer lesquelles de ces lignes passent au plus près du schéma attendu.

Dans la transformée de Hough, dite aussi transformée standard de Hough ou SHT, chaque ligne est un vecteur de coordonnées paramétriques :

- θ : l'angle
- ρ : la norme du vecteur (la longueur du segment perpendiculaire à la droite d'angle θ et passant par l'origine)

En transformant toutes les lignes possibles qui passent par un point, c'est-à-dire en calculant la valeur de ρ pour chaque θ , on obtient une sinusoïde unique appelée espace de Hough. Si les courbes associées à deux points se coupent, l'endroit où elles se coupent dans l'espace de Hough correspond aux paramètres d'une droite qui relie ces deux points.

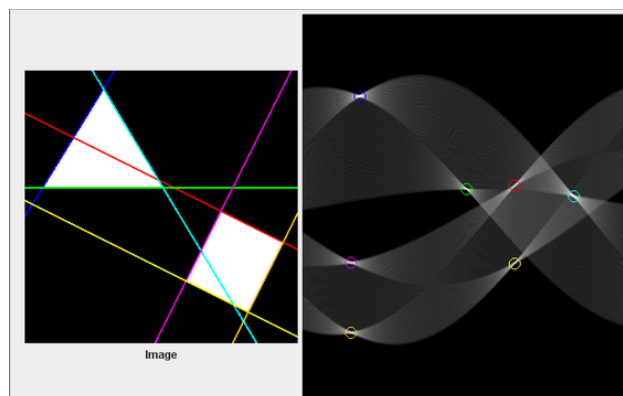


FIGURE 3.1 – Transformée de Hough

3.1.2 Codage de Freeman absolu

Codage avec un nombre limité de bits de la direction locale d'un élément de contour défini dans une image discrète, puis constitution d'une chaîne de codes à partir d'un pixel initial, considérant qu'un élément de contour relie 2 pixels connexes.

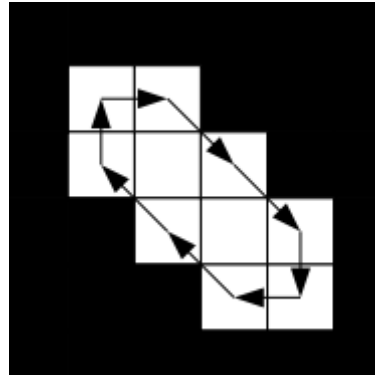


FIGURE 3.2 – Constitution d'une chaîne de codes

3.1.3 Codage de Freeman relatif

Dans cette variante on code le changement de direction plutôt que de la direction.

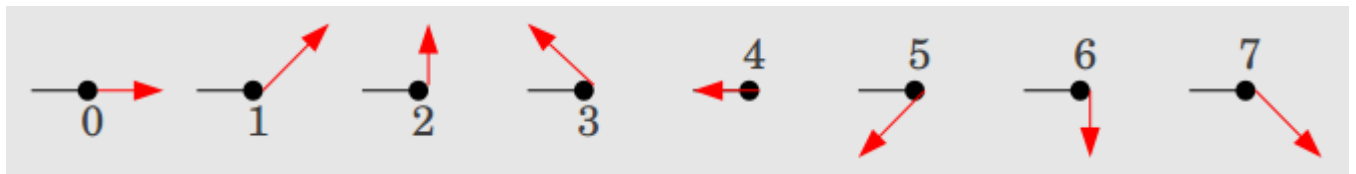


FIGURE 3.3 – Codage du changement de direction

Le code de Freeman standard est invariant en translation uniquement. Le code Freeman relatif est invariant en translation et aux rotations de 45°.

Codage sur 2 bits pour connexité 4. Codage sur 3 bits pour connexité 8. Codage sur 4 bits pour connexité 8 + longueur 2. Etc...

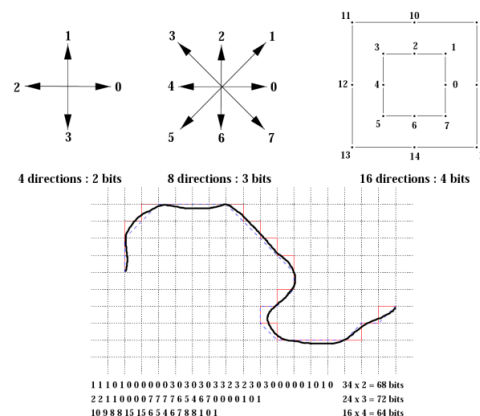


FIGURE 3.4 – Nombre de directions

3.1.4 Régression linéaire

On approche un ensemble de points par un segment de droite. Pour cela on minimise un résidu entre le modèle (la droite) et les données (points repérés par leurs coordonnées).

– Résidu :

$$d^2(a, b) =$$

3.2 Détection de couleur

CHAPITRE 4

Analyse des classes

Dans ce chapitre, nous présentons les classes `Hough.java`, `PictureHandler.java` et `HoughView` sur lesquelles nous avons dû travailler.

4.1 `Hough.java`

4.1.1 Constructeur

```
public Hough(int width, int height)
```

Points d'entrée :

- width : largeur de l'image ; entier
- height : hauteur de l'image ; entier

4.1.2 Méthodes (Transformée de Hough)

```
public void vote(int x, int y)
```

Points d'entrée :

- x : largeur de l'image/2 ; entier
- y : hauteur de l'image/2 ; entier

```
public List<double[]> getWinners(int threshold, int radius);
```

on récupère la valeur extrême de la transformée de Hough

Points d'entrée :

- threshold : seuil de l'image
- radius : rayon

Point de sortie :

- winners : tableau contenant les valeurs extrêmes de Rho et Théta ; tableau de réels

```
private int distance(int r0, int t0, int r1, int t1)
```

Points d'entrée :

- r0 : point 0 de Rho ; entier
- t0 : point 0 de Théta ; entier
- r1 : point 1 de Rho ; entier
- t1 : point 1 de Théta ; entier

Point de sortie :

- dist : Retourne la valeur minimale entre dist et le maximum de la valeur absolue entre (r0-r1) et (t0-t1) ; entier

4.1.3 Méthodes (Conversions)

public int RhoToIndex(double rho)

Points d'entrée :

– rho : réel

Point de sortie :

– On retourne un entier qui est spécialement converti pour rentrer dans notre index (matrice de valeur de Rho)

public double IndexToRho(int index)

Points d'entrée :

– index : entier

Point de sortie :

– On retourne un réel qui vient de la conversion d'un entier (Rho) de la matrice d'index.

public int ThetaToIndex(double theta)

Point d'entrée :

– theta : réel

Point de sortie :

– On retourne un entier qui est spécialement converti pour rentrer dans notre index (matrice de valeur de Theta)

public double IndexToTheta(int index)

Point d'entrée :

– index : entier

Point de sortie :

– On retourne un réel qui vient de la conversion d'un entier (Theta) de la matrice d'index.

public double[] rhotheta_to_ab(double rho, double theta) : conversion de rho et theta pour permettre son utilisation dans une équation de droite $Y=a*X+b$

Point d'entrée :

– rho : réel

– theta : réel

Point de sortie :

– a, b : réel

4.1.4 Accesseurs

public int getMaxIndexTheta()

Point de sortie :

- maxIndexTheta : entier; on récupère la valeur maximale de theta dans l'index

public int getMaxIndexRho()

Point de sortie :

- maxIndexRho : entier; on récupère la valeur maximale de rho dans l'index

public int[][] getAccumulator()

Point de sortie :

- acc : tableau 2 dimensions d'entier; on récupère ???

4.2 PictureHandler.java

4.2.1 Constructeur

public PictureHandler(PhotoFragment cxt, int callerId)

Points d'entrée :

- cxt : photo du parent; PhotoFragment
- callerId : entier

4.2.2 Méthodes

public void onPictureTaken(byte[] data, Camera camera) ; decode de l'image Bitmap

Points d'entrée :

- data : tableau d'octets
- camera : Camera

protected void onPreExecute() : pour chaque nouvelle ligne, on crée un nouvel Hash Map les contenant

protected Void doInBackground(Bitmap... pictureFile) : dans cette méthode, on charge l'image enregistré puis on exécute la transformée de Hough puis l'extraction des lignes de l'image.

Points d'entrée :

- pictureFile : type Bitmap

protected void onPostExecute(Void result)

private void doTH(Bitmap img0) : application de l'algorithme de Hough sur l'image

Points d'entrée :

- img0 : type Bitmap

private void doLinesExtraction(Bitmap img0) : permet de faire l'extraction des lignes suite à la transformée de Hough

Points d'entrée :

- img0 : type Bitmap

private void sendLinesToDrawToUiThread(HashMap<Integer, ArrayList<Point>> lines) : Permet de dessiner les lignes stockées dans le Hashmap.

Points d'entrée :

- lines : type HashMap<Integer, ArrayList<Point>> ; Un tableau qui a comme clef des entiers permettant de retrouver plus facilement les listes de points de chaque ligne précédemment stockée.

4.3 HoughView.java

CHAPITRE 5

Tests effectués

CHAPITRE 6

Démonstration

Conclusion

Projet de développement embarqué

Spécialité Informatique Industrielle
4^{ème} année
2014-2015

Reconnaissance de grille

Résumé:

Mots clefs: Apprenti :
Alexandre BILLAY, Thibault ARTUS
alexandre.billay@etu.univ-tours.fr, theskull-machine@gmail.com

Tuteur :
Yannick KERGOSIEN
yannick.kergosien@univ-tours.fr
Polytech'Tours